PrehKeyTec GmbH Technischer Support
Scheinbergweg 10
97638 Mellrichstadt
Germany
email: support@prehkeytec.de
Web: http://www.prehkeytec.com

Schnelleinstieg "Programmierung mit dem WinProgrammer"

Im Folgenden möchten wir Ihnen die Programmierung Ihrer PrehKeyTec-Tastatur, sowie die Bedienung des WinProgrammer anhand eines einfachen Beispiels erläutern.

Installieren Sie zuvor die aktuelle Version des WinProgrammer und ggf. die notwendigen Treiber. Beachten Sie bitte die Anmerkungen in der Readme-Datei.

Weiterführende Themen zur Programmierung finden Sie im Anhang dieses Handbuchs, bzw. in der umfangreichen Online-Hilfe des WinProgrammer. Sollten Sie darüber hinaus Probleme bei der Erstellung der Tastaturprogrammierung haben, so hilft Ihnen unser Support-Team gerne weiter. Am besten beschreiben Sie das Problem in einer kurzen Email, wobei Sie das verwendete Tastaturlayout (MWF-Datei) anhängen.

Beginnen wir einfach hier...

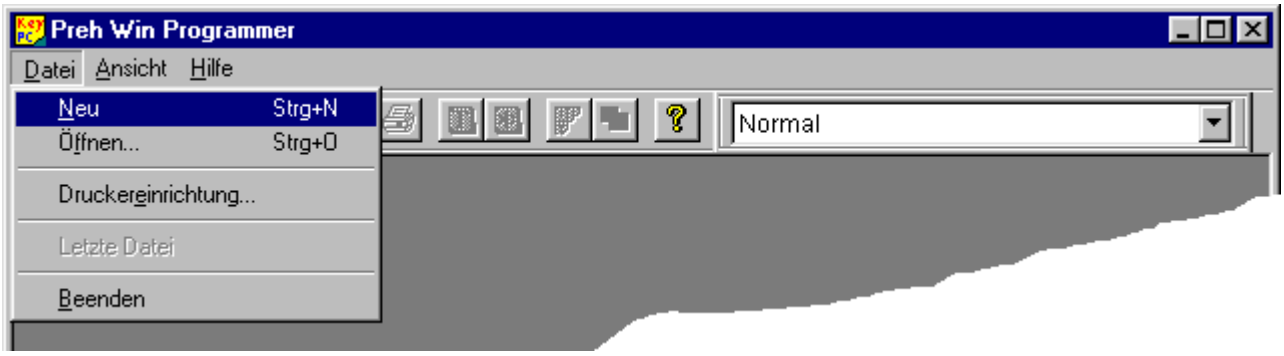


Abbildung 1

Abbildung 2

Es erscheint nun ein Dialog, wo Sie zunächst die grundlegenden Tastatureinstellungen vornehmen:

1. Tastaturgruppe auswählen:
   Die Tastaturen sind auf Registerzungen gruppiert - Alpha, Numeric oder OEM.
2. Layout der verwendeten Tastatur:
   In unserem Beispiel wählen wir die MCI 128, bei anderen Tastaturen sinngemäß.
3. Tastatursprache und CapsLock-Verhalten:
   Muß mit der Einstellung des Betriebssystems auf dem Zielrechner übereinstimmen.

Anschließend weiter mit OK.

Zusätzliche Informationen:
Während der Auswahl wird für jeden Tastaturtyp ein beispielhaftes Bild angezeigt und die Tastaturvorlage auf der Arbeitsfläche als Vorschau dargestellt.

Aktivieren Sie die Option *OPOS / JavaPOS*, wenn Sie die Preh OPOS- oder JavaPOS-Treiber einsetzen möchten. Dadurch werden die Module MSR/Keylock bereits korrekt konfiguriert.

Markieren Sie *Glidepad*, falls Sie ein solches Zeigergerät auf dem Tastenfeld integriert haben. Diese Option ist bei Alpha-Layouts sinnvoll, um hier eine angepasste Tastaturvorlage zu verwenden.

Programmieren von Standard-Tasten mittels Drag&Drop

Drag&Drop ist die einfachste Art, Standard-Tasten wie die Umschalttasten Shift, Ctrl und alle weiteren alphanumerischen Tasten zu programmieren. Hierbei verschiebt man diese Tasten mit der Maus aus einer Vorlage in das neue Layout. Dabei werden sowohl Programmierung als auch Bedruckung übernommen.

Gehen Sie hierzu wie folgt vor:

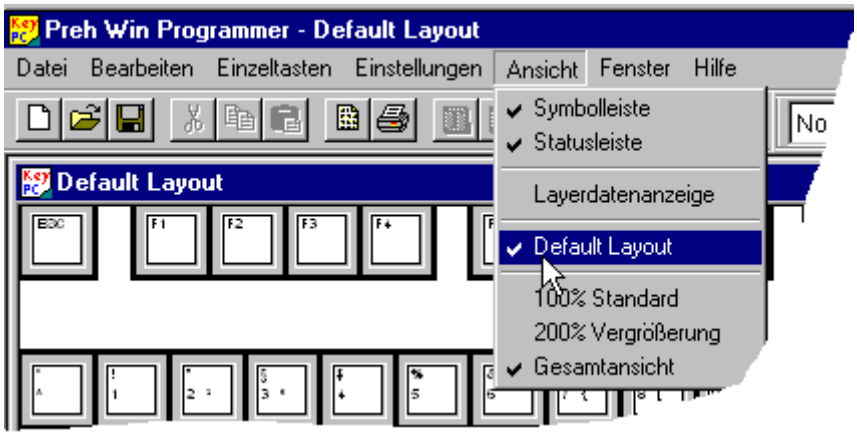


Abbildung 3

1. Aktivieren Sie die Funktion *Ansicht* → *Default Layout* um die Vorlage einzublenden.
2. Ebenso können Sie die Vorlage auch wieder ausblenden.

Der WinProgrammer öffnet diese Vorlage automatisch in der Sprache, die beim aktuell ausgewählten Layout eingestellt ist.

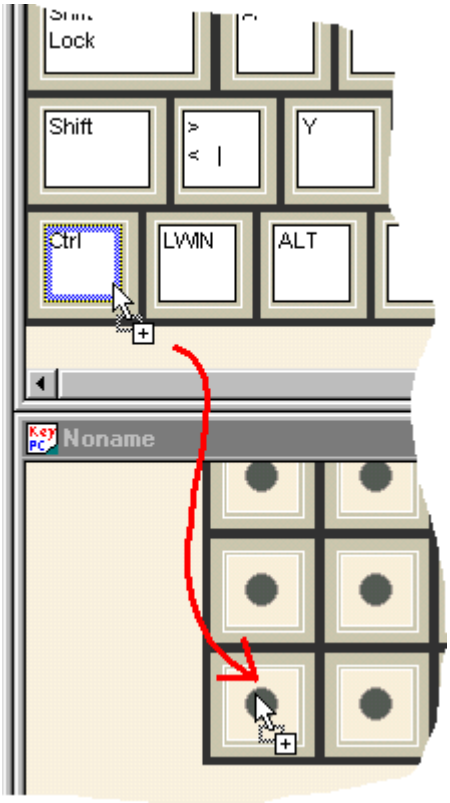


Abbildung 4

1. Markieren Sie die gewünschte Taste auf der Vorlage mittels Maus-Klick.
2. Verschieben Sie diese Taste nun bei gedrückter linker Maustaste auf die Zielposition in Ihrem Tastaturlayout.
3. Korrigieren Sie schließlich die Tastengröße über den rechten und unteren Tastenrahmen, sofern notwendig.

Anmerkungen:
Drag&Drop erkennen Sie an einem kleinen Rechteck neben dem Mauszeiger.

Wenn Sie während des Verschiebens zusätzlich die Ctrl-Taste gedrückt halten, wird kopiert, anstatt verschoben. Es erscheint ein zusätzliches + Symbol am Mauszeiger.

Es wird immer die komplette Funktionalität der Taste kopiert/verschoben - inklusive der Programmierung auf allen Layern und der Tastenbeschriftung.

In unserem Fall kopieren wir auf diese Weise:

- **die Linke Ctrl-Taste auf Position A01**
- **die Linke Shift-Taste auf Position B01**

Zählweise der Tastenpositionen

In unserem Beispiel (MCI 128) sehen Sie folgendes Tastaturlayout:

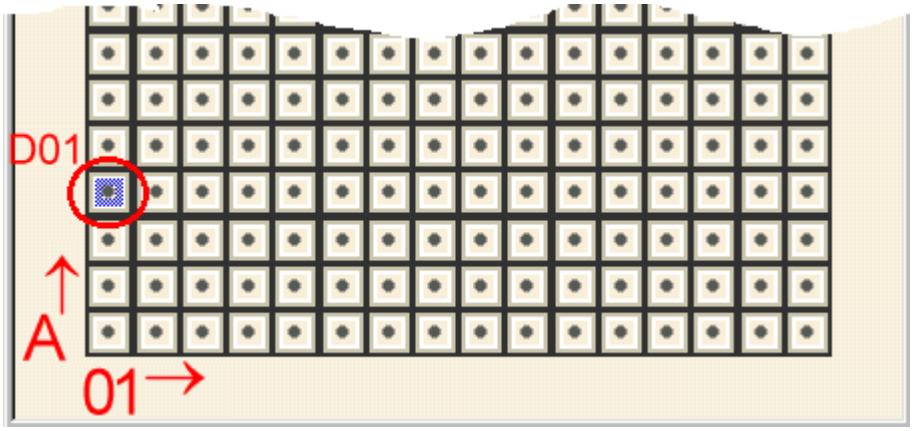


Abbildung 5

Die Tastenpositionen werden wie in Abbildung 5 dargestellt durchnumeriert. Diese erfolgt in gleicher Weise für unsere numerischen und alphanumerischen Tastaturen:

- Mittels Buchstaben (A, B, C…) links unten beginnend nach oben.
- Mittels zweistelliger Zahlen (01, 02, 03…) von links nach rechts.
- Die Tastenposition wird in der Titelleiste des Programmierdialogs angezeigt.

Programmieren unserer Beispiel-Taste D01 auf mehreren Ebenen

Auf die hervorgehobene Tastenposition D01 möchten wir nun folgende Programmierung vornehmen:

- `Normal-Layer!!!{Return}` wenn "nichts gedrückt", also kein spezieller Status aktiv ist
- `Shift-Layer!!!{Return}` bei "Shift-Status aktiv", also wenn zusammen mit Shift-Taste gedrückt
- `Control-Layer!!!{Return}` bei "Control-Status aktiv", also wenn <Control> bzw. <Strg> gedrückt

Per Doppelklick auf Tastenposition D01 erhalten wir folgenden Dialog:

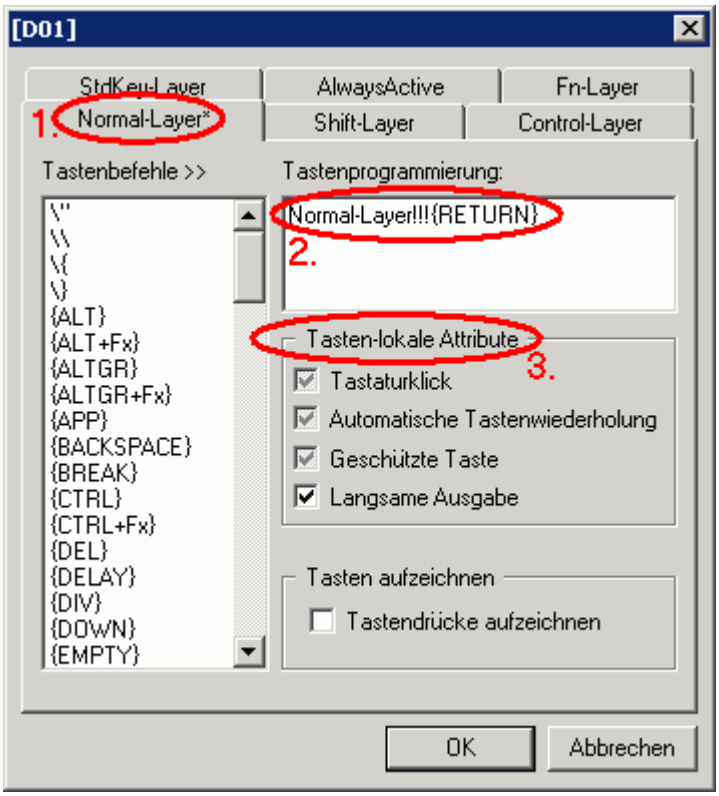


Abbildung 6

Normal-Layer:
Schritt 1: Reiter "Normal" auswählen
Schritt 2: Im Feld "Tastenprogrammierung" das eintragen, was bei "Normal aktiv" ausgegeben werden soll: `Normal Layer!!!{Return}`

Shift-Layer:
Wählen Sie nun den Reiter "Shift-Layer" aus. Wiederholen Sie Schritt 2 und tragen dort folgendes ein: `Shift-Layer!!!{Return}`

Control-Layer:
Tragen Sie nun die zugehörige Sequenz für den "Control-Layer" ein:
`Control-Layer!!!{Return}`

Zusätzlich können Sie jeweils als dritten Schritt "Tasten-lokale Attribute" vergeben. Dadurch kann man etwa festlegen, dass bei der programmierten Sequenz zusätzlich ein Klick als akustische Rückmeldung ausgegeben wird.

Den Tastenbefehl {Return} kann man entweder per Hand eingeben, oder aus der Liste "Tastenbefehle>>" auf der linken Seite auswählen.

Anmerkungen zum Dialog "Tastenprogrammierung"

Liste "Tastenbefehle":
Um spezielle Funktionen, wie in unserem Beispiel die Tastenfunktion <Return> einzugeben, doppelklicken Sie die benötigte Funktion in der Liste "Tastenbefehle". Die korrekte Schreibweise, hier {Return} wird somit in das Feld Tastenprogrammierung übertragen. Eine Liste aller unterstützen Makros und Anmerkungen zu Tastenkombinationen finden Sie im Anhang: ↳List of Supported Key Functions (Macros) auf Seite 16.

Tasten-lokale Attribute:
Zusätzlich zur eigentlichen Programmierung können Sie jeweils "Tasten-lokale Attribute" vergeben. Dadurch wird z.B. festgelegt, ob für eine Sequenz zusätzlich ein Klick als Rückmeldung ausgegeben wird.

Funktion AUSgeschaltet

Funktion EINgeschaltet

Es gelten die globalen Layerdefinitionen, wie im Menü *Einstellungen → Layereinstellungen* eingestellt.

Als Vorgabe sind die Kästchen mit einem grauen Haken versehen. Somit sind die globalen Layereinstellungen hier gültig. Um von den globalen Layereinstellungen unabhängig zu sein, empfehlen wir aber nur den Zustand EIN-, bzw. AUS zu verwenden.

Tasten aufzeichnen:
Ist diese Funktion aktiviert, werden nachfolgende Tastendrücke automatisch in das Feld *Tastenprogrammierung* eingetragen. Beispiel: 234234{F5}{TAB}{TAB}{RETURN}
Nur einzelne Tastendrücke werden aufgezeichnet – Tastenkombinationen wie {Alt+F4} müssen also manuell eingetragen werden.

Maximal 180 Zeichen (Makros / normaler Text) dürfen in das Feld *Tastenprogrammierung* eingetragen werden.

Benutzerdefinierte Layer:
Die Layer **AlwaysActive** und **Fn-Layer** werden im Kapitel ↳Benutzerdefinierte Layer "AlwaysActive" und Fn-Layer auf Seite 9 näher erläutert.

NEU: Für aktuelle Tastaturen der MCI-Familie wurde ein ↳Vereinfachtes Layerkonzept "EasyLayer" entwickelt und im WinProgrammer V2.3 eingebunden. Weitere Informationen finden Sie auf Seite 10 - bitte beachten Sie die Systemvoraussetzungen.

Wichtige Erläuterungen – Die Funktionalität des StdKey Layer

Nachfolgende Informationen sind wichtig für ein besseres Verständnis der Zusammenhänge, da der StdKey Layer eine spezielle Funktionalität besitzt:

- Mit dem Layer *StdKey* erstellen Sie eine Tastenprogrammierung, die sich dann genauso verhält, wie eine Standard-Taste auf einer normalen PC-Tastatur.
- Dies bedeutet folgendes: Wenn man auf Layer StdKey eine Tastenprogrammierung einträgt, wird immer die komplette Funktionalität dieser Taste einer Standardtastatur auf die Tastenposition ihres Preh-Keyboards abgebildet (daher auch der Name: StdKey).
- Technische Erläuterung: Ein Drücken schickt eine Standardtastatur normalerweise einen sogenannten *Make-Code* zum Rechner, beim Lösen der Taste dann den zugehörigen *Break-Code*.

Dies hat aber auch folgende wichtige Auswirkungen:

- Um die identische Funktionalität von Tasten zu erhalten, die den Status des PCs ändern (Shift, Alt, Ctrl, etc.), **müssen** diese auf dem Layer StdKey programmiert sein. Nur so funktionieren sie korrekt.
- Strings und alle Tastenkombinationen **dürfen nicht** auf dem StdKey-Layer programmiert werden - weil solche Tasten(!) auf einer "Standardtastatur" nicht enthalten sind.
- Um nicht durcheinander zu geraten, was ausgegeben werden soll, ist grundsätzlich Folgendes anzuraten: Wenn eine Taste auf dem StdKey-Layer programmiert ist, sollten hier nicht noch weitere Programmierung(en) auf einem anderen Layer platziert werden.
- Speziell bei Multilayer-Makros wie {KEY-UP} ist dies **nicht** erlaubt. Wenn Sie solche Funktionen auf verschiedenen Ebenen programmieren möchten, verwenden Sie bitte die alternativen Makros wie {Up}.

Beispiel zum manuellen Programmieren einer Taste auf dem StdKey Layer
Dies sollten Sie in unserem Beispiel auf Taste A02 programmieren:

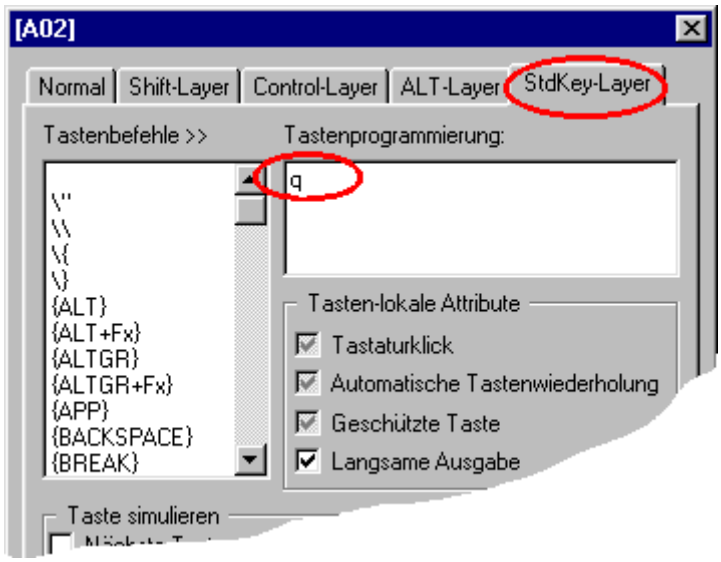


Abbildung 7

Wenn Sie auf A02 die Standard-Taste *q* wie links dargestellt programmieren, verhält sich diese Taste dann wie folgt:

- *q* wenn sonst keine Taste gedrückt
- *Q* wenn zusammen mit "Shift" gedrückt
- *@* wenn zusammen mit "AltGr" gedrückt

Weitere Ebenen müssen dazu nicht programmiert werden. Diese Taste verhält sich somit exakt so, wie auf einer Standard-Tastatur.

Voraussetzung für dieses Beispiel:
Deutscher Tastaturtreiber aktiv (z.B. keyb gr).

Nützliche Features

Anzeige → *Belegung speichern und testen* – Binäre MWX-Datei erstellen

Hiermit ist es möglich, die aktuelle Belegung vorab auf Compilierfehler testen zu lassen, ohne dass die Belegung in die PREH-Tastatur geschrieben wird.

Bei erfolgreicher Compilierung wird die Tastaturbelegung als Binärdatei / MWX-Format erzeugt – exakt so wie dieses "ausführbare Programm" in Ihre Tastatur geschrieben würde.

Das MWX-Dateiformat ist besonders zur Weitergabe an Ihre Kunden geeignet. Der Kunde erhält dann eine eindeutige binäre Belegungsdatei. Zum Download benutzt der Kunde dann Copy2MWX (DOS) oder das Download Utility C2K (Windows).

Anzeige → *Layerdatenanzeige* – Vereinfacht die Überprüfung der vorgenommenen Programmierung:

Wenn Sie diese Option einschalten, werden Tasten rosa eingefärbt, wenn sie auf dem gegenwärtig ausgewählten Layer belegt sind. Bewegen Sie den Mauszeiger über solche Tasten, wird die programmierte Sequenz angezeigt.

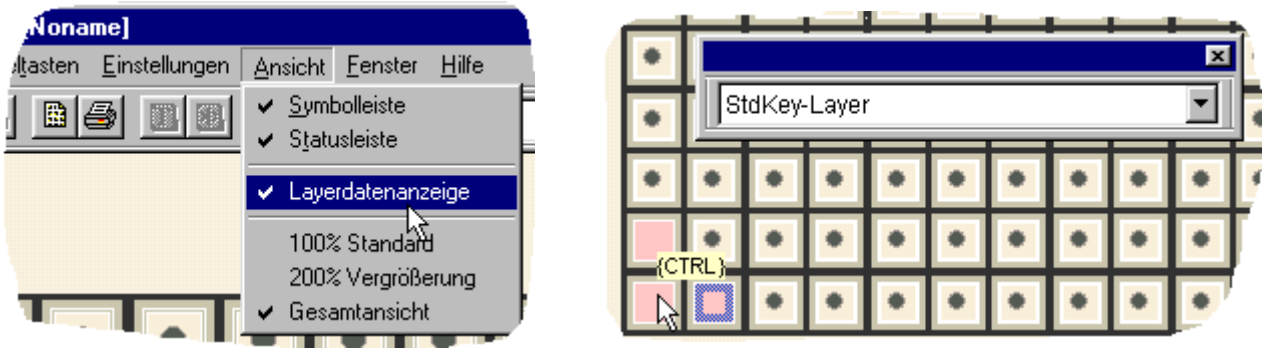


Abbildung 8

Ändern der Tastengröße

- Jede Tastenkappe - egal welche Tastengröße (1x1, 1x2, 2x2, etc.) - besitzt immer nur *eine* aktive Position.
- Im Normalfall sollten alle überdeckten Positionen gleich belegt werden, um Montagefehler auszuschließen.
- Sollen unterschiedliche Funktionen auf die überdeckten Positionen gelegt werden, darf die Taste natürlich *nicht* großgezogen werden.

- Markieren Sie die obere linke Tastenposition. Sie sehen dann einen graublauen Rahmen. Ziehen Sie diesen Rahmen nun über alle Positionen der Mehrfach-Taste.
- Beim "Großziehen" der Taste werden *nun automatisch* alle überdeckten Positionen mit der Programmierung der linken oberen Taste belegt – dies gilt für den Download und auch für das Speichern der Datei.
- Es ist somit möglich, beispielsweise eine Zweifach-Taste abzuziehen, um 180° gedreht wieder aufzusetzen und weiterzuarbeiten. Dadurch ergibt sich in diesem Beispiel die doppelte Lebensdauer, falls der Kontakt nach langer intensiver Benutzung mechanisch beschädigt sein sollte

Belegung in die Tastatur schreiben (Download)

Vor dem Download in die Tastatur empfiehlt es sich zunächst die Belegung abzuspeichern. Verwenden Sie hierzu das Menü *Datei* → *Speichern* oder *Speichern unter...*

Starten Sie nun den Download:

1. Wählen Sie Menü *Datei* → *Einstellungen an Tastatur senden*
2. Stellen Sie die Tastaturschnittstelle passend ein (siehe unten)
3. Drücken Sie OK und folgen Sie den Anweisungen, um den Download auszuführen.

Schnittstellenauswahl je nach verwendetem Tasturtyp:

- PS/2 (AT) – wenn am "normalen" Tastaturanschluss angeschlossen.
- USB – wenn Ihre PREH-Tastatur über USB angeschlossen ist.
- COM – wenn an eine serielle Schnittstelle (RS232) angeschlossen – bitte beachten, dass hierfür eine spezielle Hardwareoption benötigt wird. Darüber hinaus müssen die Parameter für den "normalen" Betrieb in folgendem Menü korrekt eingestellt sein: *Einstellungen* → *Allg.Tastatureinstellungen* → *Schnittstelle*. Diese Parameter gelten dann nach Aus- und Einschalten der Betriebsspannung. Weitere Anmerkungen hierzu finden sie im Anhang.

Bitte beachten Sie unbedingt folgende Dinge:

- Um Probleme beim Download zu vermeiden, darf während der Übertragung die Maus nicht bewegt werden.
- Ebenso sind währenddessen keine Tasteneingaben möglich.
- Wenn der Download nicht funktioniert, folgen Sie bitte den Schritten zur Problembehebung im Anhang.
- Bei PS/2 muss die Preh-Tastatur immer das erste am PC angeschlossene PS2-Gerät sein.
- Um unter Windows NT oder 2000/XP/Vista eine an PS/2 angeschlossene PREH-Tastatur programmieren zu können, muss der Preh PS/2 Tastaturtreiber korrekt installiert sein. Die jeweilige Variante wird über das Preh DriverPack installiert. Bei Windows 2000/XP/Vista muss danach im Gerätemanager/Tastaturen "Preh PS/2 Keyboard" eingetragen sein.
- Um bei einer PREH-Tastatur mit USB-Unterstützung den Download zu aktivieren, muss der PREH DriverPack korrekt installiert sein. Natürlich muss das Betriebssystem USB unterstützen und die enthaltenen HID-Geräte zuvor korrekt installiert worden sein (mindestens Win98SE bzw. Windows 2000).

Funktionstest im Texteditor

Starten Sie anschließend einen Texteditor, z.B. den Windows-Editor "Notepad" und probieren die Funktion der Taste D01 aus. Drücken Sie zusätzlich noch Shift, bzw. Control (deutsch Strg/Steuerung) dann sollte der jeweilige Text entsprechend der vorgenommenen Programmierung erscheinen.

Kommunikation prüfen

Sollte der Download der Tastaturbelegung nicht möglich sein oder die Tastatur nicht wie erwartet reagieren, sollte man zunächst die Kommunikation mit der Tastatur überprüfen. Hierzu einfach das Menü *Hilfe* → *Info über…* auswählen und den Button *Keyboard version* drücken.

Wurden alle Treiber erfolgreich installiert und die Tastatur korrekt angeschlossen, liefert die Tastatur detaillierte Informationen über die Tastaturhardware. Sollten die Tastatur keine entsprechende Ausgabe liefern, folgen Sie bitte den Schritten im Abschnitt Troubleshooting (Anhang, Seite 15).

Fortgeschrittene Programmierung

Fortgeschrittene Programmierung: Benutzerdefinierte Layer

Zum Erstellen eines benutzerdefinierten Layers, folgen Sie der nachfolgenden Anleitung. In unserem Beispiel erzeugen wir einen Layer, der immer dann aktiv ist, wenn sowohl **<Shift> UND <Ctrl>** gedrückt ist.

Wählen Sie *Einstellungen* → *Layereinstellungen*. Sie erhalten dann folgenden Dialog:

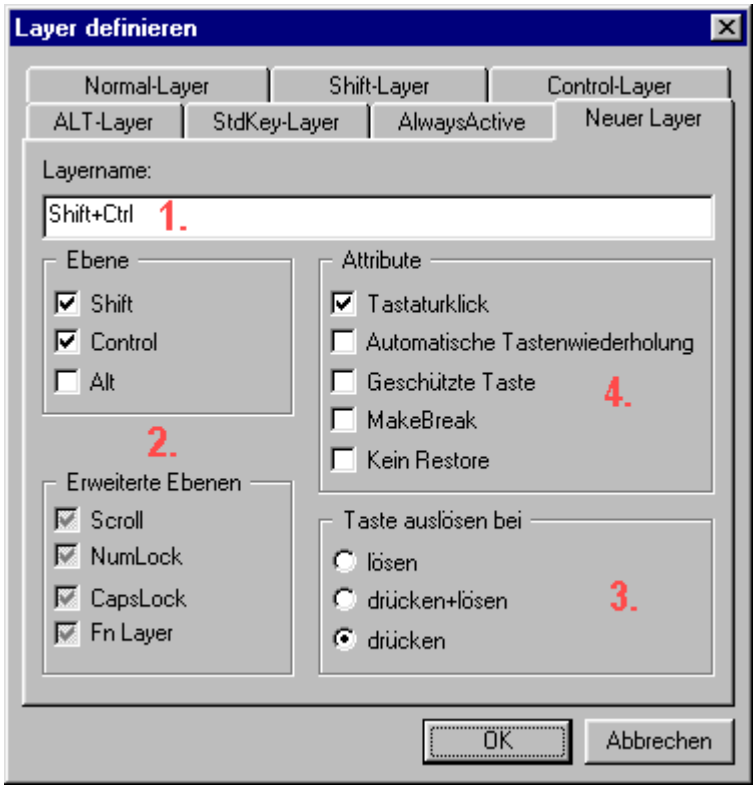


Abbildung 9

1. Wählen Sie zunächst den Reiter *Neuer Layer* aus und geben Sie dem neuen Layer einen Namen
2. Mittels *Ebene* und *Erweiterte Ebenen* legen Sie dann fest, wann der neue Layer aktiv sein soll.
3. Mit *Taste auslösen bei* legen Sie fest, zu welchem Zeitpunkt die programmierten Sequenzen jeweils ausgegeben werden sollen (üblicherweise beim Drücken der Taste)
4. Optional können Sie einstellen, dass bei diesem Layer als Voreinstellung der Tastaturklick aktiv ist.
5. Nach OK wird abgefragt, ob die Layereinstellungen abgespeichert werden sollen – Hinweis siehe unten.

Nun können Sie diesen neuen Layer genauso benutzen, wie die vordefinierten Layer Normal, Shift, usw.

Anmerkungen zum Dialog Einstellungen →Layereinstellungen:

- Die Kästchen bei Ebene / Erweiterte Ebenen haben folgende Bedeutung:
  - ☐ DARF NICHT aktiv / gedrückt sein
  - ☑ MUSS aktiv / gedrückt sein
  - ☑ "IGNORE" (Es spielt dann keine Rolle, ob diese Ebene aktiv ist oder nicht)
- Um einen benutzerdefinierten Layer wieder zu entfernen, einfach den Dialog erneut öffnen und den Layernamen löschen und mit OK bestätigen. Achtung: Alle auf diesem Layer programmierten Sequenzen werden hierbei ebenfalls gelöscht!
- Die Attribute *MakeBreak* und *Kein Restore* sollten im Normalfall nicht markiert sein. Sie sind für spezielle Funktionen reserviert.
- Eine ausführliche Beschreibung finden Sie im Index der Online-Hilfe, Stichpunkt *Layereinstellungen*

Abspeichern der Layereinstellungen

Die Layereinstellungen werden in der Startkonfiguration *preh.ini* abgespeichert. Dies ist nützlich, um benutzerdefinierte Layer nicht für jede Tastaturbelegung erneut definieren zu müssen.

Nach dem Verlassen des Dialogs mit "OK" erscheint nun ein Hinweis, ob dies in der Startkonfiguration eingetragen werden soll. Entweder werden diese Einstellungen dann abgespeichert, oder es gelten die Änderungen nur in der Aktuellen Sitzung / für die aktuelle Tastaturbelegung.

Über den Menüpunkt *Datei* → *Standard Konfiguration* kann die Startkonfiguration des WinProgrammer wieder auf „Werkszustand“ zurückgestellt werden. Bitte schließen Sie alle geöffneten Layouts, damit diese Funktion angezeigt wird.

Benutzerdefinierte Layer "AlwaysActive" und Fn-Layer

Wir haben zwei benutzerdefinierte Layer mit passenden Einstellungen vordefiniert. Alle dort abgelegten Codes werden unabhängig vom Status von Shift, Control, usw. ausgegeben – die entsprechenden Ebenendefinitionen sind bei beiden auf "ignore" eingestellt.

Beide Layer besitzen ein ähnliches Verhalten wie der Normal-Layer: Die Ausgabe der programmierten Sequenz erfolgt bereits beim Drücken der Taste. Anschließend wird der Tastaturstatus des Rechners wieder auf den alten Zustand zurückgeführt. (Restore).

AlwaysActive:
Dort abgelegte Sequenzen werden grundsätzlich immer ausgegeben, unabhängig vom Tastaturstatus. Soll nur eine Ebene verwendet werden, ist AlwaysActive deshalb dem Normal-Layer vorzuziehen.

Achtung:
Wenn auf derselben Tastenposition auch andere Layer belegt sind, wird AlwaysActive eventuell **nicht** ausgegeben. Da alle Ebenen auf "ignore" eingestellt sind, hat der Layer AlwaysActive eine sehr geringe Priorität. Layer, die "exakter" definiert sind, wie z.B. der Normal-Layer werden deshalb zuerst ausgeführt.

Fn-Layer:
Der Fn-Layer ist sehr gut geeignet, eine zweite Ebene zu programmieren. Das Fn-Attribut wird nämlich nur tastaturintern verwendet und ist somit vom Rechnerstatus unabhängig.

Einfache Makros sind zum Umschalten des Ebenenattributs verfügbar:

- {FN_ON} und {FN_OFF} schalten den Fn-Layer dauerhaft ein / aus.
- {KEY-FN} wird auf StdKey programmiert ergibt dann eine Funktionstaste wie bei einem Notebook.

Beispiel für den Einsatz des Fn-Layer:

- Sequenz auf Normal-Layer: `Testing Normal Layer{Return}`
- Sequenz auf Fn-Layer: `Testing Fn Layer{Return}{FN_OFF}`

Ergebnis:

- Unsere Beispieltaste gibt erstmal die Demo-Sequenz aus, die auf den Normal-Layer programmiert wurde.
- Sofern der Fn-Layer mittels `{FN_ON}` auf einer zweiten Taste aktiviert wurde, gibt unsere Beispieltaste dann die Sequenz des Fn-Layer aus und schaltet anschließend den Fn-Status wieder aus.

NEU *Vereinfachtes Layerkonzept "EasyLayer"*

Für die MCI-Tastaturfamilie wurde ein vereinfachtes Layerkonzept entwickelt. Dies ist ähnlich wie der Fn-Layer von den Rechnerattributen unabhängig. Es ist nun recht einfach möglich, zwischen diesen Layern zu wechseln. Bestehende Belegungen mit dem "klassischen" Layerkonzept über Shift, Control, etc. können aber weiterhin verwendet werden.

Vorteile:

- Bis zu 16 verschiedene Layer.
- Wechsel zwischen diesen Layern über Programmier-Makro. Beispiel: {EasyLayer2}
- Layerwechsel von außen über PC-Kommando steuerbar

Voraussetzung:

- MCI-Tastatur mit Firmware 605/3090 oder neuer
- WinProgrammer V2.3 (MWXC32.DLL 4.0.41.4 oder neuer)

Beispiel:

	Programmierung	Ausgabe
EasyLayer0	X	X
EasyLayer1	A	A
EasyLayer2		X
EasyLayer3	B	B
EasyLayer4		X

WinProgrammer

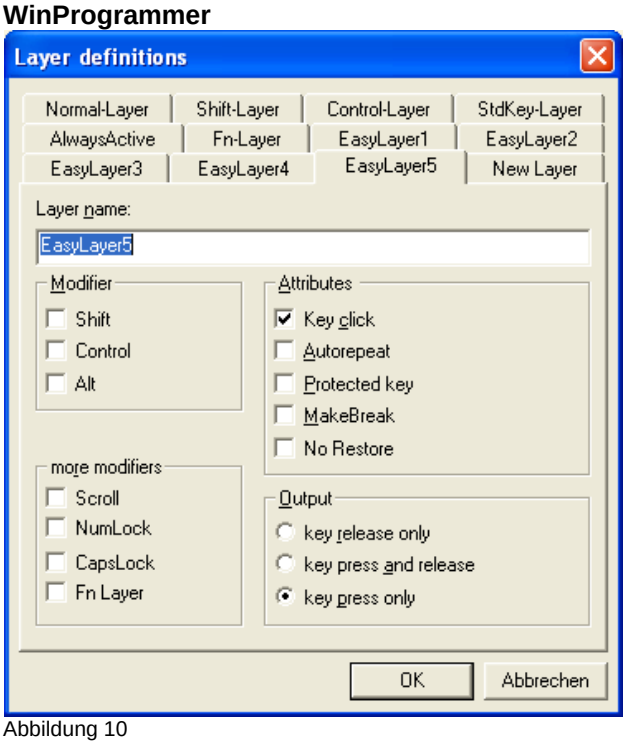


Abbildung 10

Im WinProgrammer können diese Layer durch Definition benutzerdefinierter Layer mit den Namen "EasyLayer0" bis "EasyLayer15" erzeugt werden.

Der "Easylayer0" ist immer dann aktiv, wenn keine anderen Layer zutreffen - ähnlich dem AlwaysActive Layer. Deshalb muss "Easylayer0" nicht zwingend definiert werden.

Anmerkungen:

- EasyLayer 1 ist direkt nach dem Start aktiv
- Auf dem "AlwaysActive"-Layer kann eine Default-Ausgabe programmiert werden. Dort abgelegte Codes werden ausgegeben, falls auf dem aktuell aktiven Layer nichts programmiert ist.
- In Kombination mit dem Feature "Programmierbarer Keylock" \P (siehe Seite ↳20) können nun Ebenen recht einfach per Schlüsselschalter umgeschaltet werden.
- Durch den Befehl EF 5A *XX* kann der EasyLayerXX von außen aktiviert werden (XX = 0x00..0x0F).
- Default-Layereinstellungen: Alle Belegungen schließen und dann *Datei* → *Standard Konfiguration*.

Fortgeschrittene Programmierung: Konfiguration der Module

Der nächste Schritt ist nun die Einstellung der Tastaturmodule. Folgende Tastaturmodule werden im Menü *Einstellungen* → *Module...* konfiguriert:

- MSR (Magnetkartenleser über die Tastaturleitung) – dieser ist nachfolgend beschrieben.
- Schlüsselschalter
- Barcode-Lesemodul
- Funktionskarte
- KVK Leser (alte Deutsche Krankenversichertenkarte über die Tastaturleitung)

Beispiel: Magnetkartenleser (Magnetic Stripe Reader - MSR)

Abbildung 11

- **AutoInput**

Wenn dies aktiviert ist, wird die komplette Sequenz von Daten automatisch zum Rechner übertragen, nachdem eine Magnetkarte durchgezogen wird. Wenn diese Option ausgeschaltet wird, muss die Übertragung durch ein spezielles Kommando angestoßen werden (siehe Anhang). Die Übertragung erfolgt über die Tastaturleitung.

- **Sentinels**

Jede Spur auf dem Magnetstreifen enthält so genannte Rahmendaten (Start- und Endsentinels). Über diese Checkbox können Sie festlegen, ob diese Zeichen zum Rechner übertragen werden sollen - oder nicht. Die folgende Tabelle zeigt die Codierung der Sentinels nach ISO 7811:

	Start Sentinel (SS)	End Sentinel (ES)
Spur 1	%	?
Spur 2 und 3	;	?

- **Spur 1 / Spur 2 / Spur 3**

Auswahl, welche Spuren zum Rechner übertragen werden sollen. Entfernt man eine Markierung, werden Vorspann, Kartendaten (inklusive Sentinels) und der Abspann dieser Spur unterdrückt.

- **Vorspann / Abspann**

Für jede Spur kann man einen Vor- und/oder Abspann definieren, der dann vor, bzw. nach den Spurdaten ausgegeben wird. Die Definition wird ebenso vorgenommen, wie bei einer normalen Tastenprogrammierung.

- **LRC ausgeben**

Das XOR-kodierte Prüfsummenbyte der Magnetspur kann optional ebenfalls zum Rechner übertragen werden. Wenn *LRC ausgeben* aktiviert ist, wird dieses Byte ebenso konvertiert wie die anderen Zeichen der Spur.

Die MSR-Daten werden in folgendem Format übertragen:

<Vorspann1><SS1><Data1><ES1><LRC1><Abspann1>
<Vorspann2><SS2><Data2><ES2><LRC2><Abspann2>
<Vorspann3><SS3><Data3><ES3><LRC3><Abspann3>

- **3-Spur-Leser**

Spezielle 3-Spur-Leser werden von einigen Tastaturen nicht korrekt erkannt. Deshalb sollte man diese Option bei allen Tastaturen mit 3-Spur-Leser einschalten. Sonst erscheinen die Spuren ggf. in vertauschter Reihenfolge.

- **Bad Read String**

Mittels *BadReadString* kann ein Text definiert werden, der im Falle einer fehlerhaft gelesenen Spur ausgegeben wird. Ein solcher Lesefehler kann durch eine defekte oder schmutzige Karte, Codierung nicht gemäß ISO-Standard, usw. verursacht werden. Das Token \# innerhalb des BadReadString wird bei der Ausgabe durch eine der folgenden Fehlernummern ersetzt:

0 -- Start Sentinel nicht erkannt
1 -- Paritätsfehler
2 -- Prüfsummenfehler

Um bei Preh OPOS / JavaPOS die erweiterte Fehlerauswertung auf Spurebene zu ermöglichen, ist folgender Bad Read String einzutragen: Err\#

Weitere Informationen und zusätzliche Schalter finden im Abschnitt **Special Keyboard Modes using BadReadString** (Anhang, Seite 20).

Wichtige Anmerkungen zu den Optionen:

Natürlich müssen die eingestellten Spuren auch von der **Leserhardware** unterstützt werden. Ein Leser vom Typ M1 kann Spur 1 und 2 lesen, ein M2-Leser liest Spur 2 und 3. Der M3-Leser kann alle drei Spuren lesen.

Wenn das Attribut **Langsame Ausgabe** im Vorspann eingeschaltet ist, so gilt dies auch für die folgenden Spurdaten. Die Daten werden dann nicht mit voller Geschwindigkeit zum Rechner übertragen. Ansonsten tritt möglicherweise ein Überlauf des PC-Tastaturpuffers auf, wenn der Rechner die Daten nicht schnell genug abarbeiten kann. Wir empfehlen generell, bei den MSR-Spuren *Langsame Ausgabe* einzuschalten.
Die Geschwindigkeit mit der die Zeichen bei "Langsame Ausgabe ein" zum Rechner übertragen werden, kann man im Menü *Einstellungen → Allgemeine Tastatureinstellungen → Tastenwiederholung → Langsame Tasten* anpassen. Sie sollte für auf *Mittel - Medium* eingestellt werden.
Langsame Ausgabe wird nicht ausgewertet, wenn keinerlei Programmierung im Vorspann abgelegt wurde. Es muss dann zumindest {empty} als Vorspann programmiert sein, um dies für die jeweilige Spur zu aktivieren.

Insbesondere beim MSR-Modul ist es wichtig, dass die **Tastatursprache** zum Tastaturtreiber des Betriebssystems passt. Sonst werden die Kartendaten möglicherweise nicht korrekt angezeigt!

Aufgrund der vielen verschiedenen Kombinationen ist es möglich, dass die Parameter **LRC** und **BadReadString** von einigen älteren Tastaturtypen und Magnetkartenmodulen nicht unterstützt werden!

Um die **LRC Checksumme** in Ihrer Software zu benutzen, müssen alle Spurdaten XOR kombiniert werden (inklusive Sentinels) und dann anhand der 4 least significant bits (Spur 1: 5 LSB) mit dem übermittelten LRC-Wert der Spur vergleichen.

Beispiel für eine MSR-Konfiguration:

- AutoInput: EIN, Sentinels: EIN, 3-Spur-Leser: AUS, LRC AUS
- Spur 1 aktiviert, Spur 2 und 3 deaktiviert
- Spur1 - Vorspann: msr1
- Spur1 - Abspann: end_msr1{Return}

Wenn sie nun eine Karte mit Daten auf Spur1 (DATA1 mit Sentinels % und ?) durchziehen, wird folgende Sequenz ausgegeben, mit einem Zeilenvorschub am Ende:

msr1%*DATA1***?end_msr1**

Testen der MSR-Konfiguration:
Das Testen der programmierten Sequenzen von Vorspann, Abspann, usw. sollte am besten in einem Texteditor, wie z.B. dem Windows-*Editor* oder dem DOS-Programm *Edit* erfolgen. Ziehen sie eine Karte durch den Leser und die Daten erscheinen dann wie zuvor programmiert.
Wenn falsche Zeichen (z.B. bei den Sentinels) erscheinen sollten, überprüfen Sie bitte, ob die Tastatursprache mit der im Betriebssystem eingestellten Sprache des Tastaturtreibers übereinstimmt.

Ist die Tastatur für OPOS/JavaPOS konfiguriert, ist eine entsprechende POS-Testapplikation zu verwenden. Dies ist der Werkszustand für aktuelle Tastaturen der MCI-Familie.

Anhang (in englisch)

System Requirements / Short description of the programming methods

WinProgrammer (Version 1.8 or above)

For the WinProgrammer you need an IBM AT or PS/2 compatible system (80386 or higher). The WinProgrammer runs under Windows9x, WindowsNT and Windows 2000 / XP. To enable the download, the appropriate driver for Windows NT, 2000 and XP has to be installed properly. Please see the Readme file for details about installation and usage.

Preh Programmer (PREH-MWX.EXE)

To work with the Preh Programmer you need an IBM AT or PS/2 compatible system (80286 or higher). The Preh Programmer "PREH-MWX.EXE" (Version 4.1.x and higher) can run under MS-DOS as well as under Windows 3.1, Windows95 or Windows98 in a DOS-box. Keyboards with maximum 128 key positions are supported.

For newer Windows versions (e.g. NT/2000/XP) is not possible for such DOS tools to communicate with the keyboard hardware. Nevertheless MWX files can be modified in a DOS box using Preh-MWX.EXE. For writing the files from/to the keyboard, please use the Download Utility C2K.

Download Utilities

If you want to download a previously created keytable (MWF or MWX-file) into the keyboard without using Preh Programmer or Win Programmer, you have the choice of our download utilities:

C2K (Copy to keyboard) Download Utility

If you prefer to work under Windows 9x, Windows NT, 2000 and XP use our C2K utility (Copy to keyboard). This is able to download both the MWX and MWF files. In addition it's able to read out the binary content of the keyboard in case of service. Please see the Readme file for details about usage.

Copy2mwx.exe

If you prefer to work under DOS you can use the COPY2MWX.EXE program. You can find this program in the Preh-MWX programmer package.

Syntax: `copy2mwx filename.mwx <Return>`

Try adding the parameter /w if it doesn't work in a Windows 9x DOS box:

Syntax: `copy2mwx /w filename.mwx <Return>`

Of course usage of copy2mwx.exe is also not possible in DOS box of Win2000 and WinXP.
A similar copy2mwx utility is also available for other operating systems on request.

Differences WinProgrammer – Preh-MWX (DOS)

	Win Programmer or C2K Utility	Preh-MWX.EXE or Copy2mwx.exe
MS-DOS, Windows 3.x	No	Yes [2]
Windows9x	Yes	Yes [2]
WindowsNT, 2000, XP, Vista	Yes [3]	No
OS/2	No	No [4]
Unix / Linux	No	No [4]
Read keytable	Yes [5]	Yes
Write keytable	Yes	Yes
Save keytable	Yes	Yes
Max. number of layers	128	128
Key label printing	Yes	No

[1] Online programming is only available for old MWX/MC keyboards equipped with daisy chain connector.
[2] Use new copy2mwx.exe (dated 2005) to program MCI keyboards (USB/PS2 interface) in PS2 mode. USB mode is not possible for DOS.
[3] When using PS2 interface, installation of PREH PS2 keyboard driver is required for writing keytable into the keyboard.
[4] Utility for downloading the MWX keytable file on request, installation of a special keyboard driver is required .
[5] Function is available using the "Upload" option for the C2K utility: The binary MWX keytable can be read out in case of service.

Interface settings (AT, USB, RS232)

The Preh programmable keyboards basically can be configured to run these interfaces/protocols:

- **PS/2** (AT)
- **USB** (available if keyboard is equipped with USB interface)
- **RS232** (only for MWX/MC128 family with optional factory-fitted RS232 module)

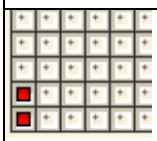
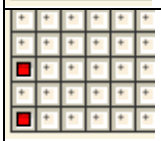
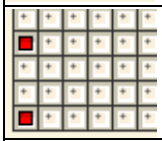
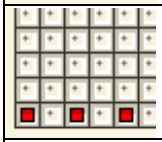
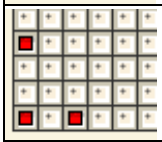
Important notes:
Of course the individual capabilities of your keyboard depend on the hardware and the cabling the keyboard is equipped with.

The computer's bios usually will display a "keyboard error" message, if the keyboard's interface setting was somehow incorrectly configured. In this case, please use one of the following key combinations to reset to the correct interface.

Special Key Combinations

Below you find some helpful key combinations for configuring and troubleshooting Preh programmable keyboards. Press and hold down one of these key combinations during powering-on the computer/keyboard. You should hold the combination for at least 5 seconds. Successful switch over is usually indicated by long beep tone(s).

Please use the appropriate **key combinations** for the Preh keyboard family you're using:

Keyboard Family / **Key combination**	**M 84/128 WX MC 25,35,80,84,128 PC-POS**	**MCI Family MC147, MC140, MF112**
 A01 + B01	PS2 (AT) protocol	Autodetect: PS/2 or USB Protocol [3]
 A01 + C01	XT (old 8086) protocol	Fixed to PS/2 Interface
 A01 + D01	RS232 protocol with Preh default parameters [1]	Fixed to USB Interface [3]
 A01 + A03 + A05	Activate a test keytable to check all key positions for electrical function. [2]	
 A01 + A03 + D01	Not supported here - Do **NOT** use! [5]	Restore the factory default keytable [4]

Notes:

[1] RS232 protocol is only available for MWX/MC with optional factory-fitted RS232 module (Default: 9600-8-O-1)

[2] Each key press and each key release should output a beep and some default key code. The stored keytable will not be changed. Please cycle power to get the keyboard back into "normal" operation.

[3] USB and Autodetect are not available for MCI keyboards with "PS2 only" electronic boards. These boards are only capable PS2 protocol.

[4] The actually programmed keytable will be replaced by the factory default keytable. Also the module settings will be reconfigured to factory defaults.

[5] This key combination is not supported for older keyboard families. It will cause them to go into RS232 mode (like A01+D01). To get back to PS2 protocol, use key combination A01+B01 instead.

Troubleshooting

Many problems are caused by loose or incorrectly connected cables. You should therefore first make sure that all cables have been properly connected. In addition you should also check any programming that you have carried out.

Problem	Possible cause	Remedy
Computer indicates "keyboard error" during start-up	• cable not correctly plugged in • cable defective • incorrect keyboard interface initialized • Timing problems between keyboard and computer	• check cable connections • replace keyboard cable • re-initialize keyboard interface • Switching off all modules which are not used via PREH WinProgrammer
Preh keyboard does not work, although the daisy-chained keyboard works	No keyboard assignment stored in the internal keyboard EEPROM	Generate keytable and download into your keyboard using the Preh WinProgrammer
Preh keyboard beeps at every key position, without displaying any characters	A fault has occurred in the transmission of the keyboard assignment table, or the contents of the EEPROM have been modified	Re-initialize keyboard interface (and download keyboard assignment table into the keyboard)
A keyboard buffer overflow occurs when transmitting long strings (e.g. MSR data)	Output speed of Preh keyboard too high.	Enable the *Slow output* attribute using the Preh WinProgrammer.
Modules do not function, or do not function correctly	Module is disabled.	Enable AutoInput for the module using the Preh WinProgrammer
Module data for MSR/Keylock is not output Notepad	Keyboard is configured to send module data via OPOS/JavaPOS channel especially in USB mode (factory default for MCI family).	Use OPOS/JavaPOS demo application for testing – or disable "OPOS Settings" using the Preh WinProgrammer.

Technical Support

- Please refer to your keyboard manual for additional information.
- Consult the Keyboard FAQ pages on the PrehKeyTec website.
- Also please check the keytable and the module settings of your keyboard.

If all the steps above did not help to solve your problem:

- Contact your local PrehKeyTec distributor in order to get technical assistance
- Contact the PrehKeyTec technical support:
  support@prehkeytec.de for worldwide support.
  techsupport@prehkeytecusa.com Support for North and South America.
- Visit the Support Area on the PrehKeyTec Website:
  http://support.prehkeytec.com

List of Supported Key Functions (Macros)

The key functions (Macros) are usually entered by just double-clicking the entry in the "Keys>>" list on the left side. You also can type them manually – then pay attention to enter them in {} (curly brackets), i.e. {F1} for the F1 key.

Some examples of key combinations: {Ctrl+F5}, {Ctrl+a}, {Delay}, {Alt+x}, {SHIFT+{ALT+F4}} ...

Important notes:

1. Key combinations using uppercase letters
For key combinations usually lowercase letters have to be used. Using uppercase letters would result to a key combination with a shifted character. See example below:

{Ctrl+A} = {Ctrl+{Shift+a}} because {Ctrl+a} ≠ {Ctrl+A}

2. Multi-Layer Macros
Multi Layer macros like {KEY-UP} are only allowed on StdKey-Layer. Of course all other layers must be left empty. Technical note: Those multi layer macros cause the key to exactly support the PS2 specified scancode sequences for such keys. On newer operating systems or especially for USB multi layer macros don't need to be used any more. Here you can use the equivalent macros like {Up} instead.

3. Programming ASCII / ANSI Codes using macro {ALTxxx}

To achieve special ASCII/ANSI characters in DOS/Windows you have to press LeftAlt key, type the character's decimal code on numeric pad, Release Alt key.

Our keyboard does the same – if key assignment is done as described below:

{Alt###} ### indicating the decimal ASCII character code.

{Alt0###} ### indicating the decimal ANSI character code.

Examples:

{Alt65} will cause the keyboard to do the Alt-Combination for a capital A (ASCII/ANSI decimal 65).
{Alt0128} will output the Windows XP Alt-Combination for the Euro sign € (ANSI decimal 128).

Available Macros	Description + Annotations
\"	Quotation mark (sign itself is reserved code – also for the key label)
\\	Backslash (sign itself is reserved code – also for the key label)
\{	Curly brackets (sign itself is reserved code – also for the key label)
\}	Curly brackets (sign itself is reserved code – also for the key label)
\^	Caret (sign itself is reserved code)
{ALT}	(left) Alt key
{ALT+Fx}	Alt + Function key (x: number 1..12)
{ALTGR}	Right ALT (AltGr) key
{ALTGR+Fx}	AltGr + Function key (x: number 1..12)
{APP}	GUI (Win) application key
{BACKSPACE}	Backspace key - abbreviation: {BS}
{BREAK}	Break key (= CTRL + Pause)
{CTRL}	(left) Ctrl key
{CTRL+Fx}	Ctrl + Function key (x: number 1..12)
{DEL}	DEL key (numeric keypad)
{DELAY}	0.5 sec output delay
{DIV}	Division key on numeric keypad
{DOWN}	Moves cursor down
{EMPTY}	Empty string
{END}	End key
{ENTER}	ENTER key
{ESC}	ESC key
{F1}	Function key F1 ... F12
{FCx}	Abbreviation for {CTRL+Fx}

{FN_OFF}	Switches Function key modifier OFF (see also Key-FN)
{FN_ON}	Switches Function key modifier ON (see also Key-FN)
{FSx}	Abbreviation for {SHIFT+Fx}
{HOME}	Home key
{INS}	Insert key
{KEY-DEL}	DEL key (multi layer macro)
{KEY-DOWN}	Cursor down (multi layer macro)
{KEY-END}	END key (multi layer macro)
{KEY-FN}	Function key modifier on/off (press/release similar to Fn key of laptop)
{KEY-HOME}	Home key (multi layer macro)
{KEY-INS}	INS key (multi layer macro)
{KEY-LEFT}	Moves cursor to the left (multi layer macro)
{KEY-N00}	Numerical block 00 key (multi layer macro)
{KEY-PGDN}	PageDown key (multi layer macro)
{KEY-PGUP}	Page Up key (multi layer macro)
{KEY-PRTSC}	Print Screen key (multi layer macro)
{KEY-RIGHT}	Cursor right (multi layer macro)
{KEY-UP}	Cursor up (multi layer macro)
{LEFT}	Cursor left
{LWIN}	Left GUI (Win) key
{MAKENUM}	NumLock key (Make-Code only; to be placed on StdKey layer)
{MAKESCROLL}	ScrollLock key (Make-Code only; to be placed on StdKey layer)
{MAKESHIFTLOCK}	CapsLock key (Make-Code only; to be placed on StdKey layer)
{MINUS}	Minus key (Numeric block)
{MUL}	Multiplication key (Numeric block)
{N.}	Delete / Dot key (Numeric block)
{N0}	Numerical block keys 0 ... 9
{NO_DATA}	Suppress the data string (only for e.g. MSR Track Headers)
{NUL}	Null byte (only for RS232 version, equivaltent to Ctrl+2)
{NUMLOCK}	NumLock key
{PAUSE}	Pause key
{PGDN}	Page Down
{PGUP}	Page Up
{PLUS}	Plus key (Numeric block)
{POSBarcode}	OPOS Barcode header / terminator
{POSFC}	OPOS Functioncard/-pen header / terminator
{POSKey001} ...128}	OPOS Key001 ... 128 scancodes 0x68 …
{POSKeylock}	OPOS Keylock header / terminator 0x65
{POSMSR1}	OPOS MSR Track1 header / terminator 0x62
{POSMSR2}	OPOS MSR Track2 header / terminator 0x63
{POSMSR3}	OPOS MSR Track3 header / terminator 0x64
{PRTSC}	Prtint Screen key
{RCTRL}	Right Ctrl key
{RESET}	Ctr + Alt + Del Macro
{RESETSTATUS}	Macro sending the release codes of both Shift, Ctrl, Alt and GUI keys
{RETURN}	RETURN key
{RIGHT}	Cursor right
{RSHIFT}	Right Shift key
{RWIN}	Right GUI (Win) key
{SCROLL-LOCK}	ScrollLock key
{SHIFT}	(Left) Shift key
{SHIFT+Fx}	SHIFT + Function key F1 ... F12
{SPACE}	Space Bar (in a string, this macro must be used at the end of a line)
{STAR}	Multiplication key (Numeric block)
{SYS}	Switches on SysRq function
{SYSBREAK}	Switches off SysRq function
{TAB}	Tab key
{UP}	Cursor up

New macros since WinProgrammer 2.3 (MWXC32.DLL V4.0.41.3):

{MEDIA_PREV}	Scan previous Track
{MEDIA_NEXT}	Scan next Track
{VOLUME_MUTE}	Mute
{LAUNCH_CALCULATOR}	Calculator
{MEDIA_PLAY_PAUSE}	Media Play/Pause
{MEDIA_STOP}	Media Stop
{VOLUME_DOWN}	Volume Down
{VOLUME_UP}	Volume Up
{BROWSER_HOME}	WWW Home
{BROWSER_SEARCH}	WWW Search
{BROWSER_FAVORITES}	WWW Favorites
{BROWSER_REFRESH}	WWW Refresh
{BROWSER_STOP}	WWW Stop
{BROWSER_FORWARD}	WWW Forward
{BROWSER_BACK}	WWW Back
{LAUNCH_EXPLORER}	My Computer
{LAUNCH_MAIL}	Mail
{LAUNCH_MEDIA}	Media Select
{POWER}	System Power
{SLEEP}	System Sleep
{WAKE}	System Wake

Special Commands for PREH Keyboards

With the following commands you can control the PREH-Keyboard and Extension Modules. For an easy implementation into your Windows application, you should use our MWX function DLL or OPOS/JavaPOS packages. This can be downloaded from our website. See the included documentation for details.

Command	Response	Parameter	Response	Function
EC	FA	<LCD data>		send data to LCD
ED	FA	<LED data>		set LED's
EF...	FA / FE			Special commands for PREH keyboards
EF 03				init default keytable
EF 05				init test table
EF 10	<ID string>			read ID string
EF 18				MSR autoinput on
EF 19				MSR autoinput off
EF 1A	<MSR data>			MSR read data
EF 1E				KL autoinput on
EF 1F				KL autoinput off
EF 20	<KL data>			KL read data
EF 21				BCR autoinput on
EF 22				BCR autoinput off
EF 23	<BCR data>			BCR read data
EF 2B	"Beep“			Keyboard Beep
EF 42	<LED Off>			Accept LED Off
EF 43	<LED Green>			Accept LED Green
EF 44	<LED Red>			Accept LED Red

Special Keyboard Modes using BadReadString

Basically the MSR module's *BadReadString* option is used to define a text which is sent instead of the track data in case of erroneous magnetic card reading.
In addition various switches can be entered in the *BadReadString* option to configure special keyboard modes. The special keyboard mode switches listed below are available for most of the new Preh keyboard families having firmware dated 2001 or newer.

Special keyboard mode switches for BadRead String:

\#	MSR: Output the Error number for faulty MSR swipe reading: 0: No start sentinel recognized, 1: Parity error, 2: Checksum error
\A	MSR: Support AAMVA
\C	MSR: Support CADL
\D	MSR: Empty Tracks: Don't output anything
\N	MSR: Don't output Sentinels for BadReadString
\S	MSR: Output Sentinels for BadReadString (using default sentinels).
\U	Universal language: Module data are output as Sequence of ALT-Combinations (e.g. A = ALT65)
\X	MSR: Error on all three tracks: Don't output anything (no data, no header, no terminator)
\R	MSR: Use old-style Caps behaviour.
\B	MSR: Beep in case of BadRead.
\W	MSR/Keylock: Special OEM data protocol (PS/2 interface only).
\L	MSR: SlowOutput activated (regardless of setting for header/terminator).
\G	MSR: Empty tracks: Mark them with an asterisk *
\Q	Keylock: Keyboard output is disabled in position 0.
\P	Keylock: Keylock codes programmable (see parameters below)

Important note:

- Not all switches are available for every Preh keyboard model. Especially older keyboard models do not support those switches at all.
- Usually the switches can be combined by entering them "in a row". Example: Err\#\A\C
- All characters must be entered into the BadReadString textbox as listed above – in *capital* letters.
- Usually you have to cycle power to activate the new parameters.

Parameters for switch \P – Programmable keylock:

- Feature only available for MCI keyboards having firmware 605/3018 or newer
- When using switch \P, you always have to enter the *complete* scancode sequence into each of the module assignments.
- Programming of the keylock is done via keylock header/terminator and other unused modules:
  0: KLH (Keylock Header)
  1: KLT (Keylock Terminator)
  2: FIH (Function card / Insert header)
  3: FIT (Function card / Insert terminator)
  4: FRH (Function card / Remove header)
- On latest MCI keyboards it's very easy to switch between layouts – using the new "EasyLayer" concept (see page 10 for details)

Copyright

© Copyright PrehKeyTec GmbH 2008

Veröffentlicht durch die PrehKeyTec GmbH
Die PrehKeyTec GmbH behält sich das Recht vor, die in diesem Handbuch beschriebenen Produkte sowie die vorliegende Publikation jederzeit ohne vorherige Ankündigung zu aktualisieren bzw. zu ändern.

Diese Bedienungsanleitung darf nicht ohne vorherige schriftliche Erlaubnis der PrehKeyTec GmbH vervielfältigt, bearbeitet und in elektronischer Form sowie in anderen Sprachen übersetzt werden.

Warenzeichen

Die in dieser Bedienungsanleitung genannten Marken- und Produktnamen sind Warenzeichen bzw. eingetragene Warenzeichen der jeweiligen Inhaber.

Beispiele: Microsoft, MS-DOS, Windows, Windows 98, Windows NT, Windows 2000, Windows XP, Windows Vista are registered trademarks of Microsoft Corporation